

Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería Licenciatura en Ciencias Computacionales

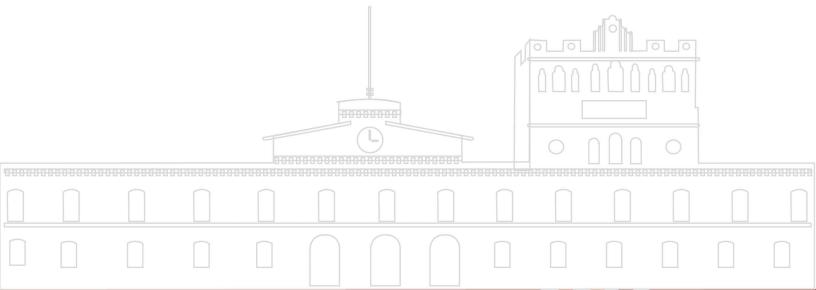
Práctica 0

Sexto semestre, Grupo dos

Catedrático: Dr. Eduardo Cornejo Velázquez

Base de datos distribuidas

Equipo: Juan Carlos Montes Gonzalez Jennifer Resendiz Isidro



Marco Teórico

Procedimientos Almacenados (Stored Procedures)

Los procedimientos almacenados son programas creados por el usuario y almacenados directamente en el servidor de la base de datos. Su objetivo principal es agrupar reglas u operaciones comerciales que deben ejecutarse de manera continua, con el fin de evitar la repetición de código en aplicaciones externas. Entre sus ventajas destacan:

- Mayor eficiencia: Al ejecutarse en el mismo motor de la base de datos.
- Facilitan el mantenimiento: Permiten la preservación y reutilización del código.
- Refuerzan la seguridad: Los usuarios pueden ejecutar procesos sin necesidad de acceder directamente a las tablas subyacentes.

Functiones (Functions)

Las funciones en SQL son similares a los procedimientos, pero se distinguen por retornar siempre un valor o un conjunto de valores (como tablas). Pueden considerarse como *vistas paramétricas*, ya que permiten generar resultados dinámicos en función de los parámetros ingresados.

Las funciones se emplean frecuentemente para:

- Simplificar consultas complejas.
- Encapsular cálculos o transformaciones comunes.
- Aumentar la modularidad en el diseño de bases de datos.

En sistemas modernos, las funciones pueden devolver valores escalares (números, texto, fecha) o incluso conjuntos de registros completos.

Estructuras de Control: Condicionales y Repetitivas

Las estructuras de control son esenciales en la programación de bases de datos, ya que permiten dirigir el flujo de ejecución de un programa, realizar elecciones basadas en condiciones y repetir fragmentos de código de forma controlada.

1.3.1 Estructuras Condicionales

IF-THEN-ELSIF-ELSE-END IF: Permite el análisis secuencial de condiciones:

- IF-THEN: Evalúa una condición: si es verdadera (TRUE), ejecuta un bloque de instrucciones.
- ELSIF: Permite verificar condiciones adicionales si la anterior no se cumple.
- ELSE: Bloque opcional que se ejecuta cuando ninguna de las condiciones anteriores es verdadera.

CASE-WHEN-THEN-ELSE-END CASE: Evalúa una expresión con múltiples valores posibles.

1.3.2 Estructuras Repetitivas (Bucles)

LOOP-EXIT WHEN-END LOOP: Ejecuta un bloque de código al menos una vez; la salida del ciclo se controla mediante EXIT o EXIT WHEN.

WHILE-LOOP-END LOOP: Evalúa la condición antes de ejecutar el bloque. Si la condición es falsa inicialmente, el bloque no se ejecuta.

FOR-IN-LOOP-END LOOP: Repite un número específico de veces dentro de un rango definido. El contador se declara implícitamente y no requiere definición en DECLARE. Además, permite la iteración en orden inverso con IN REVERSE.

Disparadores (Triggers)

Los disparadores son programas que se activan automáticamente en respuesta a eventos específicos en la base de datos, como inserciones (INSERT), actualizaciones (UPDATE) o eliminaciones (DELETE) de datos. Su objetivo es preservar la integridad y coherencia de los datos sin requerir intervención del usuario. Los disparadores son útiles para:

- Implementar reglas empresariales de forma automática.
- Auditar modificaciones en la base de datos.
- Ejecutar acciones en cascada, como añadir entradas en tablas relacionadas o verificar restricciones complejas.

En SQL, los disparadores pueden definirse para ejecutarse antes (BEFORE) o después (AFTER) del evento que los activa, y a nivel de fila o de tabla. Esta flexibilidad exige utilizarlos con precaución para evitar impactos negativos en el rendimiento.

Ejemplo 1

DELIMITER ;

```
Listing 1: Fragmento Mecanico
DELIMITER $$
CREATE TRIGGER trg_actualizar_inventario
AFTER INSERT ON insumo
FOR EACH ROW
BEGIN
    UPDATE inventario
    SET cantidad = cantidad - NEW. cantidad
    WHERE id_inventario = NEW.id_inventario;
    IF (SELECT cantidad FROM inventario WHERE id_inventario = NEW.id_inventario) < 0
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: Inventario insuficiente';
    END IF:
END $$
DELIMITER ;
Ejemplo 2
                            Listing 2: Fragmento Propietario
DELIMITER $$
CREATE PROCEDURE sp_ingresos_auto (IN p_id_auto INT)
BEGIN
    SELECT a.marca, a.modelo, SUM(r.cobro) AS total_ingresos
    FROM auto a
    JOIN ruta r ON a.id_auto = r.id_auto
    WHERE a.id_auto = p_id_auto
    GROUP BY a.marca, a.modelo;
END $$
```

Referencia

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006). Fundamentos de bases de datos (McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S.A.U., Ed.; 5th ed.). McGraw-Hill Interamericana.